Parallel computing 26.1.2018
Exercise 3

Submit each of the solutions separately to Moodle by 26.1. 09:00 (1 hour before exercises).

The following tasks are practical OpenMP parallelizations of parallel algorithms. We'll start covering OpenMP at Monday lecture. You can find examples, reference, and tutorials also from `www.openmp.org`. Now remember that the number of processors is low. Thus, there is no need to try to parallelize to more than a few processors. Sequential programs to be parallelized can be found from course www-page. Run the programs either on your own computer, a computer class computer, or the computer of the course (account and login information will be sent by email by Monday). Try to achieve some speedup!

11. Parallelize the algorithm to find a duplicate value in an integer array. The algorithm returns the first index of one such element that has a duplicate in the array. You can find a sequential $O(n^2)$ implementation at course www-page. Make a parallel OpenMP version of it. Notice that $O(n^2)$ is not optimal, but we'll ignore it this week.

12. Parallelize the algorithm to find all duplicate values in an integer array. Now the algorithm returns an array of all those values that have more than one instance in the input array. You can find a sequential $O(n^2)$ implementation at course www-page. Make a parallel OpenMP version of it. Notice that $O(n^2)$ is not optimal, but we'll ignore it this week.

13. What are the time complexity, used number of processors, and work of the following parallel procedure (as a function of N)? Which PRAM variant is required? You can assume that all processor execute in strict synchrony. $A+i*N/2$ means a subarray $A[i*N/2..]$ of array A. I.e., referring to the same array, but with different start point.

```
procedure doit(A : array; N : integer);
        if N > 1 then begin
                for i := 0 to N/2−1 pardo
                        A[i] := A[i+N/2];
                for i := 0 to 1 pardo
                        doit(A + i*N/2, N/2);
                for j:= 1 to log(N) do
                        for i := 0 to N−j pardo
                                A[i] := A[i] + A[i+j];
end     end
```

Write the following algorithms either in algorithm notation, or in English with simple and precise steps. Use PRAM notation (shared arrays), and concentrate on data.

Let us consider a voting problem where the input is a shared array of N elements, each having a random integer element $1..N$. The result of the algorithm is the integer which had the most occurrences in the array. Use the strengths of the Concurrent Write -model. Initially, don't be shy on using processors. Later, you can try to optimize processor usage a bit.

14. Write a fast algorithm for the voting problem using STRONG CRCW PRAM. What are the time complexity, work, and efficiency of your algorithm?

15. Write a fast algorithm for the voting problem using any other CRCW PRAM version than STRONG CRCW. What are the time complexity, work, and efficiency of your algorithm?